

實驗一 實驗環境建立與 Debugger 操作

第一組 0410137 劉家麟 0416324 胡安鳳

1. 實驗目的

測試實驗器材:STM32L476RG

熟悉開發環境:Ac6 STM32

2. 實驗步驟

2.1 專案的建立與程式編譯

請依照助教給的 lab1_note 教學，建立一個 STM32 eclipse project，新增一個內容如下的 main.s 程式碼並透過 debugger 觀察程式執行結果。

```
.syntax unified
.cpu cortex-m4
.thumb
.text
.global main
.equ AA, 0x55

main:
    movs r0, #AA
    movs r1, #20
    adds r2, r0, r1
L: B L
```

Q: 程式執行結束後 R2 值為多少？如何觀察？

A: 可以透過 Debugger 裡面的 Register 觀察，最後 R2 的值為 105

程式尚未開始執行前

Name	Value
General Registers	
r0	536871980
r1	1068
r2	536872044
r3	0

用 Step in 一步一步執行結束後，可觀察到 R2 為 105

Name	Value
General Registers	
r0	85
r1	20
r2	105
r3	0

2.2 變數宣告與記憶體觀察

將 main.s 修改成以下程式碼並編譯執行觀察程式執行結果，並透過 memory monitor 觀察 X 內容值變化與回答問題。

```
.syntax unified
.cpu cortex-m4
.thumb

.data
X: .word 100
str: .asciz "Hello World!"

.text
.global main
.equ AA, 0x55

main:
    ldr r1, =X
    ldr r0, [r1]
    movs r2, #AA
    adds r2, r2, r0
    str r2, [r1]

    ldr r1, =str
    ldr r2, [r1]
L: B L
```

Q1: 變數 X 與 str 的初始值是由誰在何處初始化的？

A1: X = 100 以及 str = "Hello World" 的初始化是在 data segment 進行的

.data segment

```
.data
X: .word 100
str: .asciz "Hello World!"
```

.data: 存放可寫(writeable)且初始化(initialized)的程式碼與資料

.text: 存放唯讀(read-only)的程式碼與資料

.bss: 存放未初始化的程式碼與資料

Q2: 若將 X 宣告改在 text section 對其程式執行結果會有何改變?

A2: 程式依舊可以執行，只是 X 的擺放位置由 RAM 變成 ROM

把 X 宣告在 data section，可觀察到位址在 0x20000000(RAM)

Name	Value
General Registers	
r0	100
r1	0x20000000 (Hex)
r2	85
r3	0

把 X 宣告在 text section，可觀察到位址在 0x80001f4(RAM)

Name	Value
General Registers	
r0	100
r1	0x80001f4 (Hex)
r2	185
r3	0

從 LinkerScript 可以知道 RAM 跟 ROM 擺放的位址與大小

```
/* Memories definition */
MEMORY
{
    RAM (xrw)      : ORIGIN = 0x20000000, LENGTH = 96K
    ROM (rx)       : ORIGIN = 0x80000000, LENGTH = 1024K
}
```

Q3: 程式執行完畢後 r2 內容與 str 字串在 memory 前 4 個 byte 呈現內容有何差異?

程式執行完畢後 r2 所存的內容為 0x6C6C6548

Name	Value
General Registers	
r0	100
r1	0x20000004 (Hex)
r2	0x6c6c6548 (Hex)

str 字串在 memory 前 4 個 byte 為 0x48656C6C

Address	0 - 3	4 - 7	8 - B	C - F
0000000020000000	B9000000	48656C6C	6F20576F	726C6421
0000000020000010	00000000	00000000	00000000	04030020
0000000020000020	6C030020	D4030020	00000000	00000000
0000000020000030	00000000	00000000	00000000	00000000

可觀察到 r2 內容與 str 在 memory 的前 4 個 byte，順序相反

Q4: 變數 str “Hello World!”有無其他種宣告方式? 若有請說明其中一

種。

除了用 `.asciz` 宣告外，也可以用 `.ascii` 宣告如下。

```
str: .ascii "Hello World!"
```

`.asciz`: terminated with `'\0'` (null character)

`.ascii`: isn't NUL-terminated

2.3. 簡易算數與基本記憶體指令操作

這部分實驗需要同學在 `data section` 中宣告三個 `X, Y, Z` 長度為 4byte 的變數並利用 ARM 組合語言計算以下式子，找出這些變數的 `memory address` 並觀察程式執行結果。

```
X = 5
Y = 10
X = X * 10 + Y
Z = Y - X
```

Note: 該程式需使用到算數指令 `MULS`, `ADDS`, `SUBS` 及記憶體讀寫操作指令 `LDR`, `STR`

code

```
.syntax unified
.cpu cortex-m4
.thumb

.data
X: .word 5
Y: .word 10
Z: .word 0

.text
.global main

main:
ldr r0, =X
ldr r1, [r0]
ldr r2, =Y
ldr r3, [r2]
ldr r4, =Z
ldr r5, [r4]
```

```

movs r6, #10









muls r1, r1, r6
adds r1, r1, r3
str r1, [r0]

subs r5, r3, r1
str r5, [r4]

```

L: B L

程式執行完畢後，registers 所存的值

Name	Value
▼  General Registers	
 r0	0x20000000 (Hex)
 r1	60 (Decimal)
 r2	0x20000004 (Hex)
 r3	10
 r4	0x20000008 (Hex)
 r5	-50 (Decimal)
 r6	10

r0, r2, r4 儲存的是 data X, Y, Z 的位址(address)

r1, r3, r5 則是存 data X, Y, Z 的值(value)

r6 則用來暫存要用來運算的 10



最一開始 X=5, Y=10 (initialized)

$X = X * 10 + Y = 5 * 10 + 10 = 60$

$Z = Y - X = 10 - 60 = -50$

分別在 r1, r5 運算完後，再存回 X(0x20000000), Z(0x20000008) 的位址

由 memory 觀察 0x20000000, 0x20000008 這兩個位址可觀察到相同結果，代表有成功存取

0x20000000 : 0x20000000 <Signed Integer>   New Renderings...

Address	0 - 3	4 - 7	8 - B
0000000020000000	60	10	▲ -50
0000000020000010	0	536871676	536871780
0000000020000020	0	0	0